



Class 3 - Loops

MISE Summer Programming Camp 2023



Recap of Class 2

- Comparison and boolean operators return booleans
 - Comparison operators: ==, <=, >, etc
 - Boolean operators: and, or, not
- Conditional statements
 - Use boolean expressions to determine which lines of code to run
 - if, elif, else
- Functions are a way to package code
- Variable scope
 - variables are recognized in the context they were initialized, but not recognized outside of the scope.

*Remember: == is not the same as =
x == y returns True if x is equal to y
x = y assigns the value of y to x*



Anatomy of a function

`def func(parameters):`

parameters are variables that will be provided when the function is called

`body`
`return X`

indent

*contains the actions (statements) that the function performs
returns a value (optional)*



Variable scope

```
1 def f(x):  
2     y = 5  
3     return x + y  
4  
5 print(f(4)) # Prints 9  
6 print(x) # Crashes!  
7 print(y) # Crashes!
```

Variables defined in the body of a function definition are only defined inside the indented block!

In the code on the left, the two last print statements will crash because we never defined a variable `x` or `y` in that scope.



Problem: Print every integer from 1 to 100

What if we need 1,000 print statements? 10,000?

Really tedious to write and hard to read

```
1 print(1)
2 print(2)
3 print(3)
4 print(4)
5 print(5)
6 print(6)
7 print(7)
8 print(8)
9 print(9)
```



What are loops for?

Loops allow us to run a chunk of code repeatedly until we are "done".

Loops Part I: While Loops



Solution #1: While loop

```
1 i = 1
2 while i <= 100:
3     print(i)
4     i = i + 1
```

This while loop is only 4 lines of code.



Anatomy of a while loop

while *condition:*

A boolean expression (evaluates to True or False).

Determines whether we keep looping over the while loop body or not.

body

While loop body. Runs every time expression evaluates to True. Usually multi-line

indent (commonly 2 or 4 spaces. Standardized in each codebase)

[Visualize here!](#)



Example

Line 1: Initialize the variable (i)

Line 2: While loop condition. $i \leq 100$ is a boolean expression that evaluates to either True or False

Line 3 & 4: While loop body

Line 3: Print i once every loop

Line 4: Increment the variable

```
1 i = 1
2 while i <= 100:
3     print(i)
4     i = i + 1
```

[Visualize here!](#)

What does this print?

```
1 i = 0
2 while i <= 6:
3     i = i + 2
4     print(i)
```



Continue & Break keywords

continue: skip the rest of the current iteration and move on to the next iteration.

break: skips the rest of the current iteration and exits the while loop.

```
1 i = 0
2 while i <= 6:
3     i = i + 2
4     if i == 4:
5         continue
6     print(i)
```

```
1 i = 0
2 while i <= 6:
3     i = i + 2
4     if i == 4:
5         break
6     print(i)
```



Practice Problem!

Let's use a while loop to calculate the sum of every integer from 1 to 100

Once you get some experience writing while loops, it'll be easier to solve these types of problems.

With experience, you'll learn that a good way to start is by initializing:

```
i = 1
```

```
sum = 0
```



itempool.com/mise23/live

Pop Quiz 1:

We want to use a while loop to calculate the sum of every integer from 1 to 100.

Complete the following program to do so:

```
1 i = 1
2 sum = 0
3 while ____ (a) ____:
4     sum = ____ (b) ____
5     i = ____ (c) ____
6 print(sum)
```

while loop body



Challenge: Num Digits

Given some integer n , how can we find the number of digits of that integer?

```
def num_digits(n):  
    num_digits = 0  
    while ... :  
        n = ...  
        num_digits = ...  
    return num_digits
```



Challenge: Num Digits

Given some integer n , how can we find the number of digits of that integer?

```
def num_digits(n):  
    num_digits = 0  
    while ... :  
        n = n // 10  
        num_digits = ...  
    return num_digits
```




Challenge: Num Digits

Given some integer n , how can we find the number of digits of that integer?

```
def num_digits(n):  
    num_digits = 0  
    while n >= 1:  
        n = n // 10  
        num_digits = num_digits + 1  
    return num_digits
```



Challenge: Num Digits

Does 0 have 1 or 0 digits...?

```
def num_digits(n):  
    if n == 0:  
        return 1  
    num_digits = 0  
    while n >= 1:  
        n = n // 10  
        num_digits = num_digits + 1  
    return num_digits
```



Loops Part II: For loops



Solution #2: For loop

```
1 for i in range(100):  
2     print(i)
```



Anatomy of a for loop

for *i* in *range(n)*:

Repeats an action a number of times given by the integer n

The variable i will take the value between 0 and n-1 throughout the execution

body

Runs n times. Usually multi-line

indent





Example: Printing a line of n stars

```
1 n = int(input())
2
3 for i in range(n):
4     print('*', end='')
5 print()
```

Read n from input

Repeat n times

Print a * with no line break

Print a line break



How else can we iterate in a for loop?

- 1 `for i in range(n):` *i goes from 0 to $n - 1$*
- 2 `for i in range(a, b):` *i goes from a to $b - 1$*
- 3 `for i in range(a, b+1):` *i goes from a to b*
- 4 `for i in range(a, b, 2):` *i goes from a to $b - 1$ skipping every 2*
- 5 `for i in range(a, b, k):` *i goes from a to $b - 1$ skipping every k*
- 6 `for i in range(a, b, -1):` *i goes from a to one before b , assuming $a > b$*



Pop Quiz 2:

What is the output of the following program:

```
1 for i in range(1, 4):  
2     print(2 * i, end=' ')
```




More complicated example: is a number prime?

```
1 ▾ def isPrime(n):  
2 ▾     if (n < 2):  
3 ▾         return False  
4 ▾     for factor in range(2, n):  
5 ▾         if (n % factor == 0):  
6 ▾             return False  
7     return True
```