[Note: Advanced String Slicing](https://courses.edx.org/xblock/block-v1:MITx+6.00.1x+1T2021+type@vertical+block@9f9084f6b5d746a5b370bf0e388c2697?show_title=0&show_bookmark_button=0&recheck_access=1&view=student_view&format=Finger%20Exercises)

You've seen in lecture that you can slice a string with a call such as s[i:j], which gives you a portion of string s from index i to index j-1. However this is not the only way to slice a string! If you omit the starting index, Python will assume that you wish to start your slice at index 0. If you omit the ending index, Python will assume you wish to end your slice at the end of the string. Check out this session with the Python shell:

>>> s = 'Python is Fun!'

>>> s[1:5]

'ytho'

>>> s[:5]

'Pytho'

>>> s[1:]

'ython is Fun!'

>>> s[:]

'Python is Fun!'

That last example is interesting! If you omit both the start and ending index, you see your original string!

There's one other cool thing you can do with string slicing. You can add a third parameter, k, like this: s[i:j:k]. This gives a slice of the string s from index i to index j-1, with step size k. Check out the following examples:

>>> s = 'Python is Fun!'

>>> s[1:12:2]

'yhni u'

>>> s[1:12:3]

'yoiF'

>>> s[::2]

'Pto sFn'

The last example is similar to the example s[:]. With s[::2], we're asking for the full string s (from index 0 through 13), with a step size of 2 - so we end up with every other character in s. Pretty cool!

[Note: The Python 'in' operator](https://courses.edx.org/xblock/block-v1:MITx+6.00.1x+1T2021+type@vertical+block@6b9551edbece4ee0958f1bc66dbfb1ab?show_title=0&show_bookmark_button=0&recheck_access=1&view=student_view&format=Finger%20Exercises)

The operators in and not in test for collection membership (a 'collection' refers to a string, list, tuple or dictionary - don't worry, we will cover lists, tuples and dictionaries soon!). The expression

element in coll

evaluates to True if element is a member of the collection coll, and False otherwise.

The expression

element not in coll

evaluates to True if element is **not** a member of the collection coll, and False otherwise.

Note this returns the negation of element in coll - that is, the expression element not in coll is equivalent to the expression not (element in coll).

### Exercise 1

* "a" + "bc"



* 3 \* "bc"



* "3" \* "bc"



* "abcd"[2]



* "abcd"[0:2]



* "abcd"[:2]



* "abcd"[2:]



### Exercise 2 part 1

Assume we've made the following assignments:

> str1 = 'hello'

> str2 = ','

> str3 = 'world'

1. str1





1. str1[0]





1. str1[1]





1. str1[-1]





1. len(str1)





### Exercise 2 part 2

Assume we've made the following assignments:

> str1 = 'hello'

> str2 = ','

> str3 = 'world'

[Note: The Python 'in' operator](https://courses.edx.org/xblock/block-v1:MITx+6.00.1x+1T2021+type@vertical+block@6b9551edbece4ee0958f1bc66dbfb1ab?show_title=0&show_bookmark_button=0&recheck_access=1&view=student_view&format=Finger%20Exercises)

The operators in and not in test for collection membership (a 'collection' refers to a string, list, tuple or dictionary - don't worry, we will cover lists, tuples and dictionaries soon!). The expression

element in coll

evaluates to True if element is a member of the collection coll, and False otherwise.

The expression

element not in coll

evaluates to True if element is **not** a member of the collection coll, and False otherwise.

Note this returns the negation of element in coll - that is, the expression element not in coll is equivalent to the expression not (element in coll).

1. str1[len(str1)]





1. str1 + str2 + str3





1. str1 + str2 + ' ' + str3





1. str3 \* 3





1. 'hello' == str1





### Exercise 2 part 3

Assume we've made the following assignments:

> str1 = 'hello'

> str2 = ','

> str3 = 'world'

1. 'HELLO' == str1





1. 'a' in str3





str4 = str1 + str3

‘low’ in str4





1. str3[1:3]





1. str3[:3]





Exercise 2 part 4

Assume we've made the following assignments:

> str1 = 'hello'

> str2 = ','

> str3 = 'world'

> str4 = str1 + str3

1. str3[:-1]





1. str1[1:]





1. str4[1:9]





1. str4[1:9:2]





1. str4[::-1]





### Exercise 3

[Fahrenheit system...](http://www.fahrenheittocelsius.com/)

if 6 > 7:

print("Yep")



if 6 > 7:

print("Yep")

else:

print("Nope")



var = 'Panda'

if var == "panda":

print("Cute!")

elif var == "Panda":

print("Regal!")

else:

print("Ugly...")



temp = 120

if temp > 85:

print("Hot")

elif temp > 100:

print("REALLY HOT!")

elif temp > 60:

print("Comfortable")

else:

print("Cold")



temp = 50

if temp > 85:

print("Hot")

elif temp > 100:

print("REALLY HOT!")

elif temp > 60:

print("Comfortable")

else:

print("Cold")



## Exercise 6

**Exercise 4**

[Note: What does +=, -=, \*=, /= stand for?](https://courses.edx.org/xblock/block-v1:MITx+6.00.1x+1T2021+type@vertical+block@a5f6ec8e36fb4a87bec46e793890cc03?show_title=0&show_bookmark_button=0&recheck_access=1&view=student_view&format=Finger%20Exercises)

a += b is equivalent to a = a + b

a -= b is equivalent to a = a - b

a \*= b is equivalent to a = a \* b

a /= b is equivalent to a = a / b

num = 0

while num <= 5:

print(num)

num += 1

print("Outside of loop")

print(num)



numberOfLoops = 0

numberOfApples = 2

while numberOfLoops < 10:

numberOfApples \*= 2

numberOfApples += numberOfLoops

numberOfLoops -= 1

print("Number of apples: " + str(numberOfApples))



num = 10

while num > 3:

num -= 1

print(num)



num = 10

while True:

if num < 7:

print('Breaking out of loop')

break

print(num)

num -= 1

print('Outside of loop')

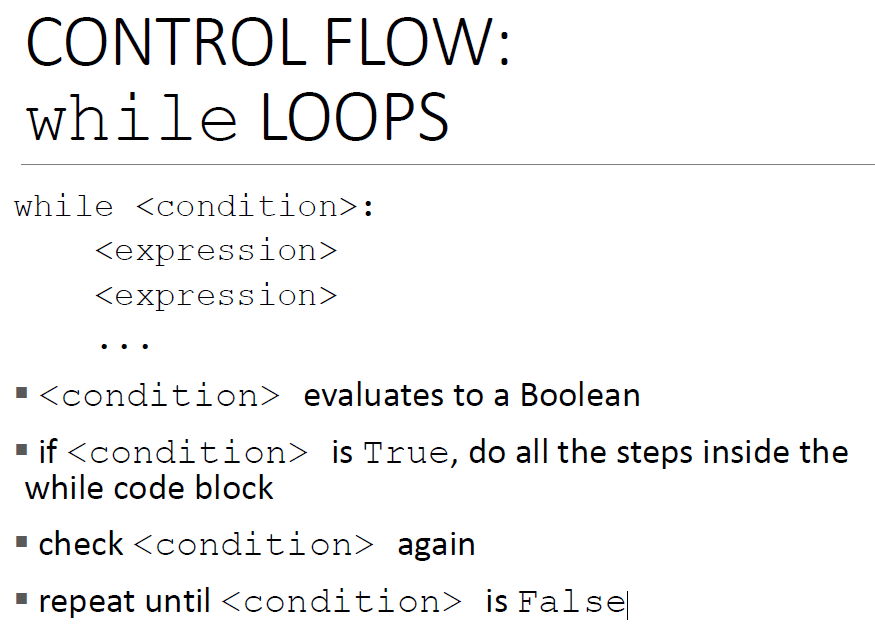
**Note:** If the command break is executed within a loop, it halts evaluation of the loop at that point and passes control to the next expression. Test some break statements inside different loops if you don't understand this concept!



1. num = 100
2. while not False:
3. if num < 0:
4. break
5. print('num is: ' + str(num))



## Exercise: **while**



1. Convert the following into code that uses a while loop.

prints 2  
prints 4  
prints 6  
prints 8  
prints 10  
prints Goodbye!

num = 0

while num < 10:

    num = num + 2

    print(num)

print("Goodbye!")

2. Convert the following into code that uses a while loop.

prints Hello!  
prints 10  
prints 8  
prints 6  
prints 4  
prints 2

num = 10

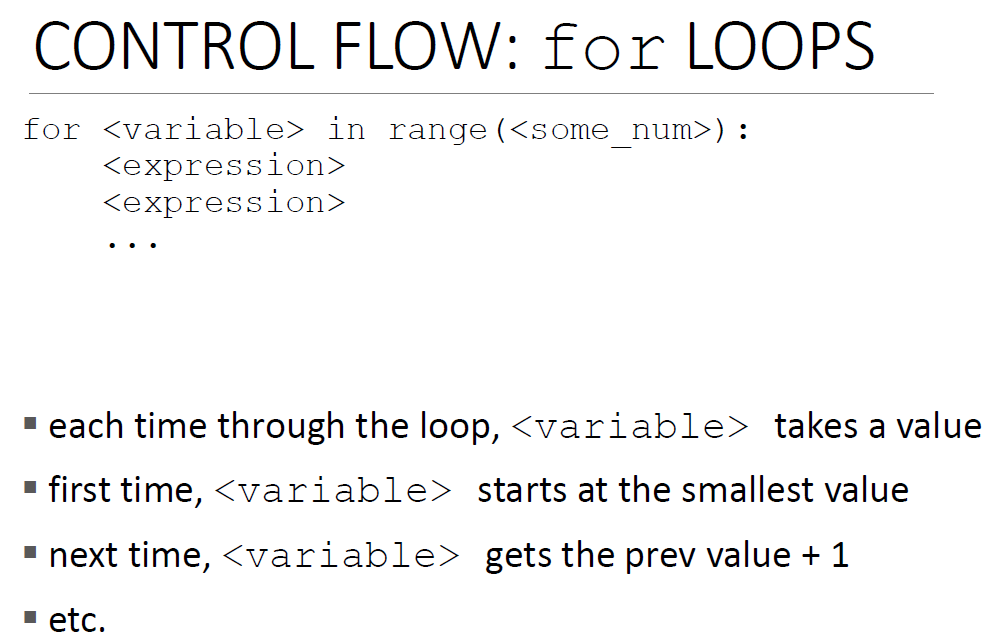
print("Hello!")

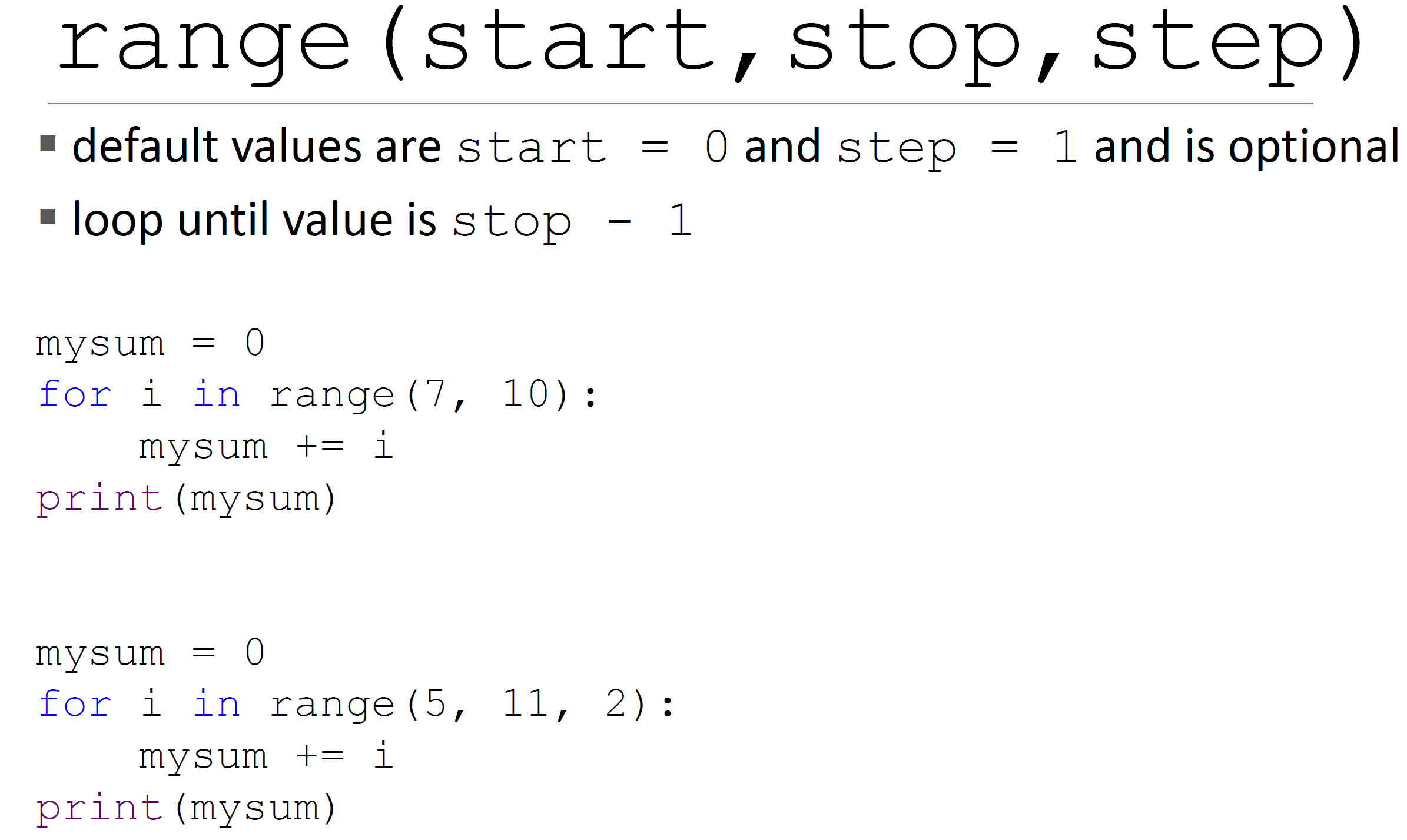
while num > 0:

    print(num)

    num = num - 2

## Exercise: **for**



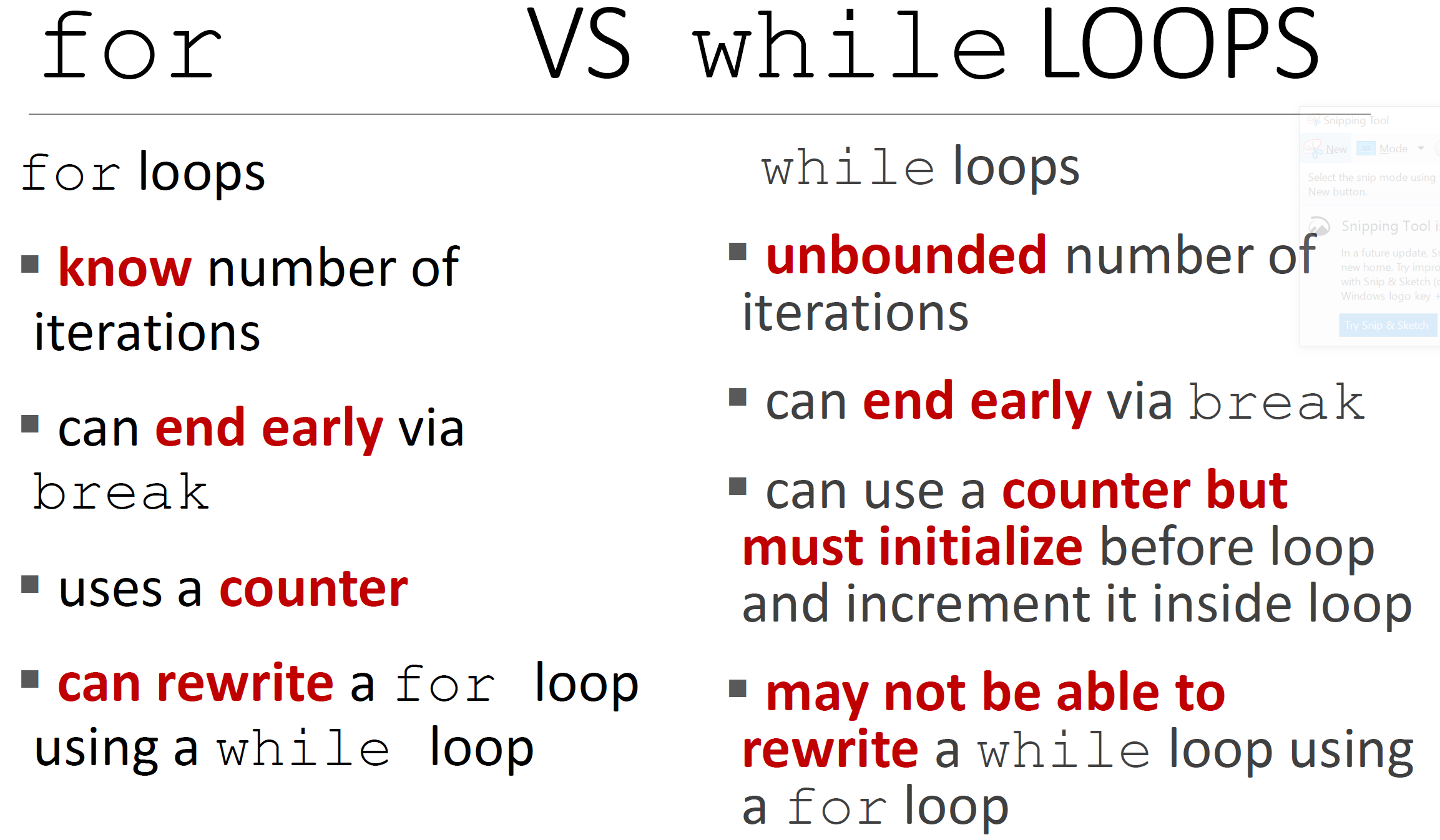


range(start,stop,step)

class range(stop: int)

range(stop) -> range object range(start, stop[, step]) -> range object

Return an object that produces a sequence of integers from start (inclusive) to stop (exclusive) by step. range(i, j) produces i, i+1, i+2, ..., j-1. start defaults to 0, and stop is omitted! range(4) produces 0, 1, 2, 3. These are exactly the valid indices for a list of 4 elements. When step is given, it specifies the increment (or decrement).



1. Convert the following code into code that uses a for loop.

prints 2  
prints 4  
prints 6  
prints 8  
prints 10  
prints Goodbye!

for num in range(2, 11, 2):

    print(num)

print("Goodbye!")

2. Convert the following code into code that uses a for loop.

prints Hello!  
prints 10  
prints 8  
prints 6  
prints 4  
prints 2

print("Hello!")

for num in range(10, 1, -2):

    print(num)

3. Write a for loop that sums the values 1 through end, inclusive. end is a variable that we define for you. So, for example, if we define end to be 6, your code should print out the result:

**21**

which is 1 + 2 + 3 + 4 + 5 + 6.

end = 6

mysum = 0

for n in range(1, end+1):

    mysum += n

print(mysum)

## Exercise 5

num = 10

for num in range(5):

print(num)

print(num)



divisor = 2

for num in range(0, 10, 2):

print(num/divisor)



for variable in range(20):

if variable % 4 == 0:

print(variable)

if variable % 16 == 0:

print('Foo!')



for letter in 'hola':

print(letter)



count = 0

for letter in 'Snow!':

print('Letter # ' + str(count) + ' is ' + str(letter))

count += 1

break

print(count)



## Exercise 6

* 1. How many times does 6 print out?



* 1. How many times does . print out?



* 1. How many times does 0 print out?



* 1. How many times does x print out?



* 1. How many times does done print out?



greeting = 'Hello!'

count = 0

for letter in greeting:

count += 1

if count % 2 == 0:

print(letter)

print(letter)

print('done')

* 1. How many times does H print out?



* 1. How many times does e print out? Disregard the letters in the word done.



* 1. How many times does l print out?



* 1. How many times does o print out? Disregard the letters in the word done.



* 1. How many times does ! print out?



* 1. How many times does done print out?



school = 'Massachusetts Institute of Technology'

numVowels = 0

numCons = 0

for char in school:

if char == 'a' or char == 'e' or char == 'i' \

or char == 'o' or char == 'u':

numVowels += 1

elif char == 'o' or char == 'M':

print(char)

else:

numCons -= 1

print('numVowels is: ' + str(numVowels))

print('numCons is: ' + str(numCons))

* 1. How many times does o print out? Disregard the o's in last two print statements.



* 1. How many times does M print out?



* 1. What will the value of the variable numVowels be?



* 1. What will the value of the variable numCons be?



### Exercise 7

5/5 points (graded)

**ESTIMATED TIME TO COMPLETE: 10 minutes**

Here is some code:

**Code Sample**

iteration = 0

count = 0

while iteration < 5:

for letter in "hello, world":

count += 1

print("Iteration " + str(iteration) + "; count is: " + str(count))

iteration += 1

**Iteration 0; count is: 12**

**Iteration 1; count is: 24**

**Iteration 2; count is: 36**

**Iteration 3; count is: 48**

**Iteration 4; count is: 60**

We wish to re-write the above code, but instead of nesting a for loop inside a while loop, we want to nest a while loop inside a for loop. Which of the following loops gives the same output as the **Code Sample**?

Try to answer the following questions by just reading the code. Reading code is a very good skill to have (and will help you both in your programming career and on your exams!). It is okay to check your answers that you obtain from just reading the code, then in your interpreter run the code for the ones you got wrong on your first attempt.

**Test 1**

for iteration in range(5):

count = 0

while True:

for letter in "hello, world":

count += 1

print("Iteration " + str(iteration) + "; count is: " + str(count))

Yes, Test 1 gives the same output as the Code Sample

No, Test 1 does not give the same output as the Code Sample

**…..**

**Iteration 0; count is: 257736**

**Iteration 0; count is: 257748**

**Iteration 0; count is: 257760**

**Test 2**

for iteration in range(5):

count = 0

while True:

for letter in "hello, world":

count += 1

print("Iteration " + str(iteration) + "; count is: " + str(count))

break

Yes, Test 2 gives the same output as the Code Sample

No, Test 2 does not give the same output as the Code Sample

**Iteration 0; count is: 12**

**Iteration 1; count is: 12**

**Iteration 2; count is: 12**

**Iteration 3; count is: 12**

**Iteration 4; count is: 12**

**Test 3**

count = 0

phrase = "hello, world"

for iteration in range(5):

index = 0

while index < len(phrase):

count += 1

index += 1

print("Iteration " + str(iteration) + "; count is: " + str(count))

Yes, Test 3 gives the same output as the Code Sample

No, Test 3 does not give the same output as the Code Sample

correct

**Test 4**

count = 0

phrase = "hello, world"

for iteration in range(5):

while True:

count += len(phrase)

break

print("Iteration " + str(iteration) + "; count is: " + str(count))

Yes, Test 4 gives the same output as the Code Sample

No, Test 4 does not give the same output as the Code Sample

correct

**Test 5**

count = 0

phrase = "hello, world"

for iteration in range(5):

count += len(phrase)

print("Iteration " + str(iteration) + "; count is: " + str(count))

Yes, Test 5 gives the same output as the Code Sample

No, Test 5 does not give the same output as the Code Sample

s = "GEEKS"

# Iterate over index and string

for element in range(0, len(s)):

    print("index:", element, "character:", s[element])

**index: 0 character: G**

**index: 1 character: E**

**index: 2 character: E**

**index: 3 character: K**

**index: 4 character: S**