1. You should be familiar with the built-in range() function that generates a sequence of integers from a start point to a stop point with an optional step increment.

range(stop) # default start=0, step=1

range(start, stop[, step])

Unfortunately, it only works with integers. In this exercise, we will create our own simple version of the built-in range() function but for floating point numbers. Create a new script 'C:\labs\gen.py' with a generator function called frange() which accepts at least two parameters (start, stop) and an optional parameter with default (step=0.25)). Be wary of the possibility of a step = zero being passed in!

frange(start, stop[, step=0.25])

Test with the following calls in main():

print(list(frange(1.1, 3)))

print(list(frange(1, 3, 0.33)))

print(list(frange(1, 3, 1))) # Should print [1.0, 2.0].

print(list(frange(3, 1))) # Should print an empty list.

print(list(frange(1, 3, 0))) # Should print an empty list.

print(list(frange(-1, -0.5, 0.1)))

print(frange(1, 2)) # Should print <generator object frange at 0x....>

for num in frange(3.142, 12):

Stretch

2. Modify your 'c:\labs\gen.py' script and enhance your frange() function so that it can accept only one parameter that acts as the stop of the sequence. The start parameter should then default to 0, with step set to 0.25.

Test the function with the following code in main():

result1 = list(frange(0, 3.5, 0.25))

result2 = list(frange(3.5))

if result1 == result2:

print("Default worked!")

else:

print(f"Oops! {result1} not equal to {result2}")

3. Did you notice any inaccuracies in the numbers in the previous exercises? This is expected and is the nature of using floating points. They are so serious that this code should not be used in a production environment. A solution is to use the decimal module from the Python Standard library. This has a decimal. Decimal class constructor that converts integers/strings to Decimal objects - do this for all the parameters (start, stop and step) AND then convert the value to be yielded back to a float.

Modify your script 'C:\labs\gen.py' and frange() function to use the decimal module. The test results should be more sensible! Test using same calls in question 4.