ANA-522-OL1 Spring 2022
Mod01 Week02 HW: Python Part Two
Due: Sunday January 23rd at midnight

The first part is a tutorial about handling exceptions caused by a zero denominator in divison operations. It then experimented with the case by creating a function for its purpose. You do not have to work on this one.

See Errors and Exception Handling in the text p.77 - 80 for further examples.

0.0.1 Creating functions for repeated operations

[1]:
```
# We are trying to solve: Is number X divisible by number Y
# The idea is to use remainder (mod) operator: X%Y
# and, decide if the remainder value is zero (that is divisible)
X =12
Y = 3
if X%Y == 0:
    print('12 is divisible by 3')
else:
    print('12 is Not divisible by 3')

X =12
Y = 9
if X%Y == 0:
    print('12 is divisible by 9')
else:
    print('12 is Not divisible by 9')
```

```
12 is divisible by 3
12 is Not divisible by 9
```

[2]:
```
# In the case that we encounter the same question frequently,
# it is better to make it a function, so that we can reuse it

def isDivisible(a,b):
    if a%b == 0:
        return True
    else:
        return False

aNumber = 12
for i in range(1, aNumber+1):     # i ranges from 1 to 12
    if isDivisible(aNumber,i):    # call the function to get answer
        print('{0:d} is divisible by {1:d}'.format(aNumber,i))
    else:
        print('{0:d} is NOT divisible by {1:d}'.format(aNumber,i))
```

```
12 is divisible by 1
12 is divisible by 2
12 is divisible by 3
12 is divisible by 4
12 is NOT divisible by 5
12 is divisible by 6
12 is NOT divisible by 7
12 is NOT divisible by 8
12 is NOT divisible by 9
12 is NOT divisible by 10
12 is NOT divisible by 11
12 is divisible by 12
```

[3]:
```
# If we are not careful, divisor can be zero at times
# Mathematically it is impossible to carry that out
# therefore, the ZeroDivisionError

x=12
y=0
if isDivisible(x,y):
    print('{0:d} is divisible by {1:d}'.format(x,y))
else:
    print('{0:d} is not divisible by {1:d}'.format(x,y))
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-3-231fc0bf4fcd> in <module>
      5 x=12
      6 y=0
----> 7 if isDivisible(x,y):
      8     print('{0:d} is divisible by {1:d}'.format(x,y))
      9 else:

<ipython-input-2-0f0fb053bce3> in isDivisible(a, b)
      3
      4 def isDivisible(a,b):
----> 5     if a%b == 0:
      6         return True
      7     else:

ZeroDivisionError: integer division or modulo by zero
```

[4]:

```
# One way to fix it, is to make sure that would not happen
# the remainder operation can not be carried out when the divisor is zero.
# To differentiate, we create a new function of version 2

def isDivisible2(a,b):
    if b!=0:
        if a%b == 0:
            return True
        else:
            return False
    else:
        return False

x=12
y=0
if isDivisible2(x,y):
    print('{0:d} is divisible by {1:d}'.format(x,y))
else:
    print('{0:d} is not divisible by {1:d}'.format(x,y))
```

```
12 is not divisible by 0
```

[5]:

```
# Another way to put it, is to use the original function
# but with an additional control statement as filter
def isDivisible2(a,b):
    if b!=0:
       return isDivisible(a,b)
    else:
        return False

x=12
y=0
if isDivisible2(x,y):
    print('{0:d} is divisible by {1:d}'.format(x,y))
else:
    print('{0:d} is not divisible by {1:d}'.format(x,y))
```

```
12 is not divisible by 0
```

1 The HW Problem: The Sieve of Eratosthenes

The Sieve of Eratosthenes is a technique that was developed more than 2,000 years ago to easily find all of the prime numbers between 2 and some limit, say 100. A description of the algorithm follows:

```
Write down all of the numbers from 0 to the limit
 Cross out 0 and 1 because they are not prime

   Set p equal to 2
     While p is less than the limit do
           Cross out all multiples of p (but not p itself)
           Set p equal to the next number in the list that is not crossed out
```

Report all of the numbers that have not been crossed out as prime

The key to this algorithm is that it is relatively easy to cross out every nth number on a piece of paper. This is also an easy task for a computer—a for loop can simulate this behavior when a third parameter is provided to the range function. When a number is crossed out, we knowthat it is no longer prime, but it still occupies space on the piece of paper, and must still be considered when computing later prime numbers.

As a result, you should not simulate crossing out a number by removing it from the list. Instead, you should simulate crossing out a number by replacing it with 0. Then, once the algorithm completes, all of the non-zero values in the list are prime.

Read the Python program in the following that uses this algorithm to display all of the prime numbers between 2 and a limit,say 1000.

[6]:
```python
limit = 1_000     # set the limit

numbers = []      # use numbers list to create a list with numbers from 1 to limit
for i in range(0, limit+1):
    numbers.append(i)

numbers[1] = 0    # cross out 0

p=2               # implicitly cross out 1
while p < limit:
    for i in range(p*2,limit+1,p):
        numbers[i]=0         # cross out multiple of p

    p = p + 1
    while p < limit and numbers[p]==0:
        p = p + 1            # try next number which had not been crossed out

# the final list of all zeros and prime numbers
print(numbers)
```

```
[0, 0, 2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19, 0, 0, 0, 23, 0,
0, 0, 0, 0, 29, 0, 31, 0, 0, 0, 0, 0, 37, 0, 0, 0, 41, 0, 43, 0, 0, 0, 47, 0, 0,
0, 0, 0, 53, 0, 0, 0, 0, 0, 59, 0, 61, 0, 0, 0, 0, 0, 67, 0, 0, 0, 71, 0, 73, 0,
0, 0, 0, 0, 79, 0, 0, 0, 83, 0, 0, 0, 0, 0, 89, 0, 0, 0, 0, 0, 0, 0, 97, 0, 0,
0, 101, 0, 103, 0, 0, 0, 107, 0, 109, 0, 0, 0, 113, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 127, 0, 0, 0, 131, 0, 0, 0, 0, 0, 137, 0, 139, 0, 0, 0, 0, 0, 0, 0,
0, 0, 149, 0, 151, 0, 0, 0, 0, 0, 157, 0, 0, 0, 0, 0, 163, 0, 0, 0, 167, 0, 0,
0, 0, 0, 173, 0, 0, 0, 0, 0, 179, 0, 181, 0, 0, 0, 0, 0, 0, 0, 0, 0, 191, 0,
193, 0, 0, 0, 197, 0, 199, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 211, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 223, 0, 0, 0, 227, 0, 229, 0, 0, 0, 233, 0, 0, 0, 0, 0, 239,
0, 241, 0, 0, 0, 0, 0, 0, 0, 0, 0, 251, 0, 0, 0, 0, 0, 257, 0, 0, 0, 0, 0, 263,
0, 0, 0, 0, 0, 269, 0, 271, 0, 0, 0, 0, 0, 277, 0, 0, 0, 281, 0, 283, 0, 0, 0,
0, 0, 0, 0, 0, 0, 293, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 307, 0, 0, 0, 311,
0, 313, 0, 0, 0, 317, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 331, 0, 0, 0, 0, 0,
337, 0, 0, 0, 0, 0, 0, 0, 0, 0, 347, 0, 349, 0, 0, 0, 353, 0, 0, 0, 0, 0, 359,
0, 0, 0, 0, 0, 0, 0, 367, 0, 0, 0, 0, 0, 373, 0, 0, 0, 0, 0, 379, 0, 0, 0, 383,
0, 0, 0, 0, 0, 389, 0, 0, 0, 0, 0, 0, 0, 397, 0, 0, 0, 401, 0, 0, 0, 0, 0, 0, 0,
409, 0, 0, 0, 0, 0, 0, 0, 0, 0, 419, 0, 421, 0, 0, 0, 0, 0, 0, 0, 0, 0, 431, 0,
433, 0, 0, 0, 0, 0, 439, 0, 0, 0, 443, 0, 0, 0, 0, 0, 449, 0, 0, 0, 0, 0, 0, 0,
457, 0, 0, 0, 461, 0, 463, 0, 0, 0, 467, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 479,
0, 0, 0, 0, 0, 0, 0, 487, 0, 0, 0, 491, 0, 0, 0, 0, 0, 0, 0, 499, 0, 0, 0, 503,
0, 0, 0, 0, 0, 509, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 521, 0, 523, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 541, 0, 0, 0, 0, 0, 547, 0, 0, 0, 0, 0, 0,
0, 0, 0, 557, 0, 0, 0, 0, 0, 563, 0, 0, 0, 0, 0, 569, 0, 571, 0, 0, 0, 0, 0,
577, 0, 0, 0, 0, 0, 0, 0, 0, 0, 587, 0, 0, 0, 0, 0, 593, 0, 0, 0, 0, 0, 599, 0,
601, 0, 0, 0, 0, 0, 607, 0, 0, 0, 0, 0, 613, 0, 0, 0, 617, 0, 619, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 631, 0, 0, 0, 0, 0, 0, 0, 0, 0, 641, 0, 643, 0, 0, 0, 647,
0, 0, 0, 0, 0, 653, 0, 0, 0, 0, 0, 659, 0, 661, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
673, 0, 0, 0, 677, 0, 0, 0, 0, 0, 683, 0, 0, 0, 0, 0, 0, 0, 691, 0, 0, 0, 0, 0,
0, 0, 0, 0, 701, 0, 0, 0, 0, 0, 0, 0, 709, 0, 0, 0, 0, 0, 0, 0, 0, 0, 719, 0, 0,
0, 0, 0, 0, 0, 727, 0, 0, 0, 0, 0, 733, 0, 0, 0, 0, 0, 739, 0, 0, 0, 743, 0, 0,
0, 0, 0, 0, 0, 751, 0, 0, 0, 0, 0, 757, 0, 0, 0, 761, 0, 0, 0, 0, 0, 0, 0, 769,
0, 0, 0, 773, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 787, 0, 0, 0, 0, 0, 0, 0,
0, 0, 797, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 809, 0, 811, 0, 0, 0, 0, 0, 0, 0, 0,
0, 821, 0, 823, 0, 0, 0, 827, 0, 829, 0, 0, 0, 0, 0, 0, 0, 0, 0, 839, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 853, 0, 0, 0, 857, 0, 859, 0, 0, 0, 863, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 877, 0, 0, 0, 881, 0, 883, 0, 0, 0, 887, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 907, 0, 0, 0, 911, 0, 0, 0, 0,
0, 0, 0, 919, 0, 0, 0, 0, 0, 0, 0, 0, 0, 929, 0, 0, 0, 0, 0, 0, 0, 937, 0, 0, 0,
941, 0, 0, 0, 0, 0, 947, 0, 0, 0, 0, 0, 953, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 967, 0, 0, 0, 971, 0, 0, 0, 0, 0, 977, 0, 0, 0, 0, 0, 983, 0, 0, 0, 0, 0, 0,
0, 991, 0, 0, 0, 0, 0, 997, 0, 0, 0]
```

1.1 (a) Use list comprehension to remove zeros on the list, so that the list becomes :

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

1.2 (b) Create a function that would return how many prime numbers are there in the range of [a,b]. The program should ask user for the values of a and b, the printout the answers.

For example:

```
Please give value a:
30
Please give value b:
50
There are 5 prime numbers in between 30 and 50.

Please give value a:
130
Please give value b:
150
There are 4 prime numbers in between 130 and 150.
```